```c
#include<stdio.h>
#include<stdlib.h>

#define eps 5e-5



int mem = 0, memMax = 0;
FILE *fm;
unsigned short print;



struct QDD
{
    struct no *n;
    unsigned short nqbit;
    struct lista *l;
};

struct inicio
{
    struct no *n;
};

struct meio
{
    unsigned short classe, nivel;
    struct no *el, *th;
};

struct fim
{
    float re, im;
};

struct no
{
    unsigned short tipo;
    struct lista *l;
    union atributos
    {
        struct inicio i;
        struct meio m;
        struct fim f;
    }at;
};

struct lista
{
    struct lista *l;
    struct no *n;
};

struct apply
{
    struct no    *n, *n1, *n2;
    struct apply *a, *a1, *a2;
};



typedef struct QDD QDD;

typedef struct no no;

```

```
 67  typedef struct lista lista;
 68
 69  typedef struct inicio inicio;
 70
 71  typedef struct meio meio;
 72
 73  typedef struct fim fim;
 74
 75  typedef struct apply apply;
 76
 77  typedef unsigned short Short;
 78
 79
 80
 81  void aumenta_memoria(int m)
 82  {
 83      mem += m;
 84      if(print)
 85      fprintf(fm,"\nMemUP: %d\t\t\t %d",mem,m);
 86      if(memMax<mem)
 87      {
 88          memMax = mem;
 89          if(print)
 90          fprintf(fm,"\t\tMemMax");
 91      }
 92  }
 93
 94  void diminui_memoria(int m)
 95  {
 96      mem -= m;
 97      if(mem<0)
 98      {
 99          printf("\n\nERRO MEM");
100          exit(EXIT_FAILURE);
101      }
102      if(print)
103      fprintf(fm,"\n\tMemDOWN: %d\t\t%d",mem,-m);
104  }
105
106
107
108  QDD* cria_QDD()
109  {
110      QDD *Q;
111      Q = malloc(sizeof(QDD));
112      if(Q == NULL)
113      {
114          printf("\n\nERRO QDD");
115          exit(EXIT_FAILURE);
116      }
117      aumenta_memoria(sizeof(QDD));
118      Q->n = NULL;
119      Q->l = NULL;
120      return Q;
121  }
122
123  no* cria_no_inicio()
124  {
125      no *n;
126      n = malloc(sizeof(no));
127      if(n == NULL)
128      {
129          printf("\n\nERRO INICIO");
130          exit(EXIT_FAILURE);
131      }
132      aumenta_memoria(sizeof(no));
```

```
133      n->tipo = 0;
134      n->l = NULL;
135
136      inicio i;
137      i.n = NULL;
138      n->at.i = i;
139
140      return n;
141  }
142
143  no* cria_no_meio(Short classe, Short nivel)
144  {
145      no *n;
146      n = malloc(sizeof(no));
147      if(n == NULL)
148      {
149          printf("\n\nERRO INICIO");
150          exit(EXIT_FAILURE);
151      }
152      aumenta_memoria(sizeof(no));
153      n->tipo = 1;
154      n->l = NULL;
155
156      meio m;
157      m.classe = classe;
158      m.nivel = nivel;
159      m.el = NULL;
160      m.th = NULL;
161      n->at.m = m;
162
163      return n;
164  }
165
166  no* cria_no_fim(float re,float im)
167  {
168      no *n;
169      n = malloc(sizeof(no));
170      if(n == NULL)
171      {
172          printf("\n\nERRO INICIO");
173          exit(EXIT_FAILURE);
174      }
175      aumenta_memoria(sizeof(no));
176      n->tipo = 2;
177      n->l = NULL;
178
179      fim f;
180      f.re = re;
181      f.im = im;
182      n->at.f = f;
183
184      return n;
185  }
186
187  lista* cria_no_lista()
188  {
189      lista *l;
190      l = malloc(sizeof(lista));
191      if(l == NULL)
192      {
193          printf("\n\nERRO LISTA");
194          exit(EXIT_FAILURE);
195      }
196      aumenta_memoria(sizeof(lista));
197      if(l == NULL)
198          exit(0);
```

```
199      l->l = NULL;
200      l->n = NULL;
201      return l;
202  }
203
204  apply* cria_apply()
205  {
206      apply *a;
207      a = malloc(sizeof(apply));
208
209      a->n = NULL;
210      a->n1 = NULL;
211      a->n2 = NULL;
212
213      a->a = NULL;
214      a->a1 = NULL;
215      a->a2 = NULL;
216
217      return a;
218  }
219
220
221
222  void libera_no_QDD(QDD *Q)
223  {
224      diminui_memoria(sizeof(QDD));
225      free(Q);
226  }
227
228  void libera_no(no *n)
229  {
230      diminui_memoria(sizeof(no));
231      free(n);
232  }
233
234  void libera_no_lista(lista *l)
235  {
236      diminui_memoria(sizeof(lista));
237      free(l);
238  }
239
240  void libera_lista(lista *l)
241  {
242      lista *lc;
243      while(l != NULL)
244      {
245          lc = l->l;
246          libera_no_lista(l);
247          l = lc;
248      }
249  }
250
251
252
253  lista* enlista_QDD(QDD *Q)
254  {
255      /**  l é a lista final                          ***
256      ***  la é a pilha de nós para adicionar a l  **/
257      lista *l, *la, *lc, *lf;
258      no *n;
259
260      l = cria_no_lista();
261      la = cria_no_lista();
262      la->n = Q->n;
263      lf = l;
264
```

```
265     while(la != NULL)
266     {
267         n = la->n;
268         /**  Verifica se o no n ká está na lista  **/
269         for(lc = l; lc != NULL; lc = lc->l)
270             if(lc->n == n)
271                 break;
272         if(lc == NULL)
273         {
274             /**  caso não esteja  **/
275             lf->l = cria_no_lista();
276             lf = lf->l;
277             lf->n = n;
278 
279             if(n->tipo == 1)
280             {
281                 /**  caso tenha filhos  **/
282                 la->n = n->at.m.th;
283                 lc = cria_no_lista();
284                 lc->n = n->at.m.el;
285                 lc->l = la;
286                 la = lc;
287             }
288         }
289         else
290         {
291             /**  caso já esteja  */
292             lc = la->l;
293             libera_no_lista(la);
294             la = lc;
295         }
296     }
297     l->n = Q->n->l->n;
298     return l;
299 }
300 
301 
302 
303 void mostra_lista(lista *l)
304 {
305     lista *lc;
306     Short ligacao = 0;
307     lc = l;
308     for(lc = l; lc != NULL; lc = lc->l)
309     {
310         printf("\tLigacao %u: %d\n",ligacao,lc->n);
311         ligacao++;
312     }
313 }
314 
315 void mostra_no(no *n)
316 {
317     printf("\nEndereco: %d\n",n);
318     if(n->l != NULL)
319     {
320         printf("Ligacoes anteriores:\n");
321         mostra_lista(n->l);
322     }
323     printf("Tipo");
324     switch(n->tipo)
325     {
326         case 0:
327         printf(": Inicio\n");
328         printf("Ligacoes posteriores\n");
329         printf("\tn: %d\n",n->at.i.n);
330         break;
```

```
331
332         case 1:
333         printf("/nivel: ");
334         switch(n->at.m.classe)
335         {
336             case 0:
337             printf("V");
338             break;
339
340             case 1:
341             printf("R");
342             break;
343
344             case 2:
345             printf("C");
346             break;
347         }
348
349         printf("%d\n",n->at.m.nivel);
350         printf("Ligacoes posteriores\n");
351         printf("\telse: %d\n",n->at.m.el);
352         printf("\tThen: %d\n",n->at.m.th);
353         break;
354
355         case 2:
356         printf(": Numero\n");
357         printf("%f %f",n->at.f.re,n->at.f.im);
358         break;
359     }
360     printf("\n");
361 }
362
363 void mostra_lista_com_no(lista *l)
364 {
365     lista *lc;
366     Short ligacao = 0;
367     lc = l;
368     printf("\n");
369     for(lc = l; lc != NULL; lc = lc->l)
370     {
371         printf("\tLigacao %u: %d\n",ligacao,lc->n);
372         mostra_no(lc->n);
373         ligacao++;
374     }
375 }
376
377 void mostra_arvore_ineficiente(no *n)
378 {
379     if(n == NULL)
380         return;
381     mostra_no(n);
382     mostra_arvore_ineficiente(n->at.m.el);
383     mostra_arvore_ineficiente(n->at.m.th);
384 }
385
386 void mostra_QDD(QDD *Q)
387 {
388     lista *l;
389     l = enlista_QDD(Q);
390     printf("NQBIT: %d\n",Q->nqbit);
391     mostra_lista_com_no(l);
392     printf("\n");
393     mostra_lista(Q->l);
394     libera_lista(l);
395 }
396
```

```
397
398
399 void fmostra_lista(FILE *fp, lista *l)
400 {
401     lista *lc;
402     Short ligacao = 0;
403     lc = l;
404     for(lc = l; lc != NULL; lc = lc->l)
405     {
406         fprintf(fp,"\tLigacao %u: %d\n",ligacao,lc->n);
407         ligacao++;
408     }
409 }
410
411 void fmostra_no(FILE *fp, no *n)
412 {
413     fprintf(fp,"\nEndereco: %d\n",n);
414     if(n->l != NULL)
415     {
416         fprintf(fp,"Ligacoes anteriores:\n");
417         mostra_lista(n->l);
418     }
419     fprintf(fp,"Tipo");
420     switch(n->tipo)
421     {
422         case 0:
423         fprintf(fp,": Inicio\n");
424         fprintf(fp,"Ligacoes posteriores\n");
425         fprintf(fp,"\tn: %d",n->at.i.n);
426         break;
427
428         case 1:
429         fprintf(fp,"/nivel: ");
430         switch(n->at.m.nivel)
431         {
432             case 0:
433             fprintf(fp,"V");
434             break;
435
436             case 1:
437             fprintf(fp,"R");
438             break;
439
440             case 2:
441             fprintf(fp,"V");
442             break;
443         }
444
445         fprintf(fp,"%d\n",n->at.m.nivel);
446         fprintf(fp,"Ligacoes posteriores\n");
447         fprintf(fp,"\telse: %d\n",n->at.m.el);
448         fprintf(fp,"\tThen: %d\n",n->at.m.th);
449         break;
450
451         case 2:
452         fprintf(fp,": Numero\n");
453         fprintf(fp,"%f %f",n->at.f.re,n->at.f.im);
454         break;
455     }
456     fprintf(fp,"\n");
457 }
458
459 void fmostra_lista_com_no(FILE *fp, lista *l)
460 {
461     lista *lc;
462     Short ligacao = 0;
```

```
463      lc = l;
464      fprintf(fp,"\n");
465      for(lc = l; lc != NULL; lc = lc->l)
466      {
467          fprintf(fp,"\n\tLigacao %u: %d\n",ligacao,lc->n);
468          fmostra_no(fp,lc->n);
469          ligacao++;
470      }
471  }
472
473  void fmostra_QDD(FILE *fp,QDD *Q)
474  {
475
476      lista *l;
477      l = enlista_QDD(Q);
478      fprintf(fp,"NQBIT: %d\n",Q->nqbit);
479      fmostra_lista_com_no(fp,l);
480      fprintf(fp,"\n");
481      fmostra_lista(fp,Q->l);
482      libera_lista(l);
483  }
484
485  void fmostra_QDD_sozinho(QDD *Q, char *nome)
486  {
487      FILE *fp;
488      fp = fopen(nome,"w");
489      fmostra_QDD(fp,Q);
490      fclose(fp);
491  }
492
493
494
495  void conecta_UM(no *n1, no *n2, Short lado)
496  {
497      lista *l;
498
499      switch(lado)
500      {
501          case 0:
502          n1->at.i.n = n2;
503          break;
504
505          case 1:
506          n1->at.m.el = n2;
507          break;
508
509          case 2:
510          n1->at.m.th = n2;
511          break;
512      }
513
514      l = cria_no_lista();
515      l->n = n1;
516      l->l = n2->l;
517      n2->l = l;
518  }
519
520  void conecta_DOIS(no *n, no *el, no *th)
521  {
522      conecta_UM(n,el,1);
523      conecta_UM(n,th,2);
524  }
525
526  Short desconecta_UM(no *n1, no *n2)
527  {
528      lista *l, *lc, *laux;
```

```
529     Short lado;
530     if(n1->tipo == 0)
531     {
532         n1->at.i.n = NULL;
533         lado = 0;
534     }
535     else
536     {
537         if(n1->at.m.el == n2)
538         {
539             n1->at.m.el = NULL;
540             lado = 1;
541         }
542         else
543         {
544             n1->at.m.th = NULL;
545             lado = 2;
546         }
547     }
548 
549     l = cria_no_lista();
550     l->l = n2->l;
551     for(lc = l; lc->l != NULL; lc = lc->l)
552     {
553         if(lc->l->n == n1)
554         {
555             laux = lc->l;
556             lc->l = laux->l;
557             libera_no_lista(laux);
558             break;
559         }
560     }
561     n2->l = l->l;
562     libera_no_lista(l);
563     return lado;
564 }
565 
566 void desconecta_DOIS(no *n)
567 {
568     switch(n->tipo)
569     {
570         case 0:
571         desconecta_UM(n,n->at.i.n);
572         break;
573 
574         case 1:
575         desconecta_UM(n,n->at.m.el);
576         desconecta_UM(n,n->at.m.th);
577         break;
578     }
579 }
580 
581 void transfere_conexao(no *n1, no *n2)
582 {
583     no *n;
584     Short lado;
585     while(n2->l != NULL)
586     {
587         n = n2->l->n;
588         lado = desconecta_UM(n,n2);
589         conecta_UM(n,n1,lado);
590     }
591 }
592 
593 
594 
```

```
595  void libera_QDD(QDD *Q)
596  {
597      lista *l, *lc;
598      l = enlista_QDD(Q);
599      no *n1, *n2;
600      for(lc = l; lc != NULL; lc = lc->l)
601      {
602          n1 = lc->n;
603          while(n1->l != NULL)
604          {
605              n2 = n1->l->n;
606              desconecta_UM(n2,n1);
607          }
608      }
609
610      for(lc = l; lc != NULL; lc = lc->l)
611          libera_no(lc->n);
612
613      libera_lista(l);
614      libera_lista(Q->l);
615
616      libera_no_QDD(Q);
617  }
618
619
620
621  no* copia_no(no *n1)
622  {
623      no *n2;
624      n2 = NULL;
625      switch(n1->tipo)
626      {
627          case 0:
628          n2 = cria_no_inicio();
629          break;
630
631          case 1:
632          n2 = cria_no_meio(n1->at.m.classe,n1->at.m.nivel);
633          break;
634
635          case 2:
636          n2 = cria_no_fim(n1->at.f.re,n1->at.f.im);
637          break;
638      }
639      return n2;
640  }
641
642  Short compara_no_meio(no *n1, no *n2)
643  {
644      if(n1             != n2       )
645      if(n1->tipo       == n2->tipo )
646      if(n1->at.m.nivel == n2->at.m.nivel)
647      if(n1->at.m.el    == n2->at.m.el   )
648      if(n1->at.m.th    == n2->at.m.th   )
649          return 1;
650      return 0;
651  }
652
653  Short compara_no_fim(no *n1, no *n2)
654  {
655      float re, im;
656      re = (n1->at.f.re)-(n2->at.f.re);
657      im = (n1->at.f.im)-(n2->at.f.im);
658
659      if(re>-eps)
660      if(re< eps)
```

```
661     if(im>-eps)
662     if(im< eps)
663         return 1;
664     return 0;
665 }
666
667 Short compara_apply(apply *a1, apply *a2)
668 {
669     if(a1->n1 == a2->n1)
670     if(a1->n2 == a2->n2)
671         return 1;
672     return 0;
673 }
674
675 no* produto_complexo(no *n1, no *n2)
676 {
677     no *n;
678     float re, im;
679     re = (n1->at.f.re)*(n2->at.f.re)-(n1->at.f.im)*(n2->at.f.im);
680     im = (n1->at.f.re)*(n2->at.f.im)+(n1->at.f.im)*(n2->at.f.re);
681     n = cria_no_fim(re,im);
682     return n;
683 }
684
685 no* produto_complexo_conjugado(no *n1, no *n2)
686 {
687     no *n;
688     float re, im;
689     re = (n1->at.f.re)*(n2->at.f.re)+(n1->at.f.im)*(n2->at.f.im);
690     im = (n1->at.f.re)*(n2->at.f.im)-(n1->at.f.im)*(n2->at.f.re);
691     n = cria_no_fim(re,im);
692     return n;
693 }
694
695
696
697 lista* copia_lista(lista *l1)
698 {
699     lista *l2, *lc, *lc2;
700     l2 = cria_no_lista();
701     lc2 = l2;
702     for(lc = l1; lc != NULL; lc = lc->l)
703     {
704         lc2->l = cria_no_lista();
705         lc2 = lc2->l;
706         lc2->n = lc->n;
707     }
708     return l2;
709 }
710
711 lista* copia_lista_sem_cabeca(lista *l1)
712 {
713     lista *l2, *laux;
714     l2 = copia_lista(l1);
715     laux = l2;
716     l2 = laux->l;
717     libera_no_lista(laux);
718     return l2;
719 }
720
721 void reduz_lista(lista *l)
722 {
723     no *n1, *n2;
724     lista *lc1, *lc2, *laux;
725     lc1 = l;
726     for(lc1 = l; lc1->l != NULL; lc1 = lc1->l)
```

```
727     {
728         n1 = lc1->n;
729         lc2 = lc1;
730         while(lc2->l != NULL)
731         {
732             n2 = lc2->l->n;
733             if(compara_no_fim(n1,n2))
734             {
735                 transfere_conexao(n1,n2);
736                 laux = lc2->l;
737                 lc2->l = laux->l;
738                 libera_no_lista(laux);
739                 libera_no(n2);
740             }
741             else
742                 lc2 = lc2->l;
743         }
744         if(lc1->l == NULL)
745             break;
746     }
747 }
748
749
750
751 void completa_QDD_matriz(no *n, Short r, Short c, Short exp, Short **M, lista **L)
752 {
753     no *el, *th;
754     Short ind1, ind2;
755     if((n->at.m.classe == 2)&&(exp == 1))
756     {
757         ind1 = M[r][c];
758         ind2 = M[r][c+1];
759
760         el = L[ind1]->n;
761         th = L[ind2]->n;
762
763         conecta_DOIS(n,el,th);
764     }
765     else
766     {
767         if(n->at.m.classe == 1)
768         {
769             completa_QDD_matriz(n->at.m.el,r,c,exp,M,L);
770             completa_QDD_matriz(n->at.m.th,r+exp,c,exp,M,L);
771         }
772         if(n->at.m.classe == 2)
773         {
774             completa_QDD_matriz(n->at.m.el,r,c,exp/2,M,L);
775             completa_QDD_matriz(n->at.m.th,r,c+exp,exp/2,M,L);
776         }
777     }
778 }
779
780 QDD* le_matriz(char *nome)
781 {
782     Short i, j, k;
783
784     FILE *fp;
785     fp = fopen(nome,"r");
786
787     Short N1, N2, N3;
788     fscanf(fp,"%hu %hu\n%hu\n",&N1, &N2, &N3);
789
790
791     lista **L;
792     float re, im;
```

```
793       L = malloc(N3*sizeof(lista*));
794       if(L == NULL)
795       {
796           printf("\n\nERRO L");
797           exit(EXIT_FAILURE);
798       }
799       aumenta_memoria(N3*sizeof(lista*));
800       for(i=0; i<N3; i++)
801       {
802           L[i] = cria_no_lista();
803           fscanf(fp,"%f %f",&re, &im);
804           L[i]->n = cria_no_fim(re,im);
805       }
806       fscanf(fp,"\n");
807       for(i=0; i<N3-1; i++)
808           L[i]->l = L[i+1];
809
810       Short **M;
811       M = malloc(N2*sizeof(Short*));
812       if(M == NULL)
813       {
814           printf("\n\nERRO M");
815           exit(EXIT_FAILURE);
816       }
817       aumenta_memoria(N2*sizeof(Short*));
818       for(i=0; i<N2; i++)
819       {
820           M[i] = malloc(N2*sizeof(Short));
821           if(M[i] == NULL)
822           {
823               printf("\n\nERRO M[%d]",i);
824               exit(EXIT_FAILURE);
825           }
826           aumenta_memoria(N2*sizeof(Short));
827           for(j=0; j<N2; j++)
828               fscanf(fp,"%hu",&M[i][j]);
829       }
830
831       no **N;
832       N = malloc((N2*N2-1)*sizeof(no*));
833       if(N == NULL)
834       {
835           printf("\n\nERRO N");
836           exit(EXIT_FAILURE);
837       }
838       aumenta_memoria((N2*N2-1)*sizeof(no*));
839
840       Short exp, ind;
841       exp = 1;
842       ind = 0;
843       for(i=0; i<N1; i++)
844       {
845           for(j=1; j<=2; j++)
846           {
847               for(k=0; k<exp; k++)
848               {
849                   N[ind] = cria_no_meio(j,i);
850                   ind++;
851               }
852               exp *= 2;
853           }
854       }
855
856       for(i=0; i<(N2*N2-1)/2; i++)
857           conecta_DOIS(N[i],N[2*i+1],N[2*i+2]);
858
```

```
    completa_QDD_matriz(N[0],0,0,N2/2,M,L);

    QDD *Q;
    Q = cria_QDD();
    Q->n = cria_no_inicio();
    conecta_UM(Q->n,N[0],0);
    Q->n = N[0];
    Q->nqbit = N1;
    Q->l = L[0];

    free(N);
    diminui_memoria((N2*N2-1)*sizeof(no*));
    free(L);
    diminui_memoria(N3*sizeof(lista*));
    for(i=0; i<N2; i++)
    {
        free(M[i]);
        diminui_memoria(N2*sizeof(Short));
    }
    free(M);
    diminui_memoria(N2*sizeof(Short*));
    free(fp);

    return Q;
}

void reduz_QDD(QDD *Q)
{
    no *n1, *n2, *nc;
    lista *l, *laux, *lnc1, *lnc2, *lr, *lrc, *lf;
    Short mudou, saida, regra;
    l = copia_lista(Q->l);
    while(l->l != NULL)
    {
        nc = l->l->n;
        do
        {
            mudou = 0;
            for(regra = 2; regra >= 1; regra--)
            {
                do
                {
                    saida = 0;
                    for(lnc1 = nc->l; lnc1->l != NULL; lnc1 = lnc1->l)
                    {
                        n1 = lnc1->n;
                        for(lnc2 = lnc1->l; lnc2 != NULL; lnc2 = lnc2->l)
                        {
                            n2 = lnc2->n;
                            switch(regra)
                            {
                                case 1:
                                if(n1 == n2)
                                {
                                    desconecta_DOIS(n2);
                                    transfere_conexao(nc,n2);
                                    libera_no(n2);

                                    mudou = 1;
                                    saida = 1;
                                }
                                break;

                                case 2:
                                if(compara_no_meio(n1,n2))
                                {
```

```
                                    lr = cria_no_lista();
                                    lr->n = n1;
                                    lrc = cria_no_lista();
                                    lrc->n = n2;
                                    lr->l = lrc;

                                    while(lnc2->l != NULL)
                                    {
                                        lnc2 = lnc2->l;
                                        n2 = lnc2->n;

                                        if(compara_no_meio(n1,n2))
                                        {
                                            if(n1->at.m.el == n1->at.m.th)
                                            {
                                                for(lrc = lr; lrc->l != NULL; lrc =
lrc->l)

                                                    if(lrc->l->n == n2)
                                                        break;
                                                if(lrc->l != NULL)
                                                    continue;
                                            }
                                            lrc->l = cria_no_lista();
                                            lrc = lrc->l;
                                            lrc->n = n2;
                                        }
                                    }

                                    while(lr->l != NULL)
                                    {
                                        lrc = lr->l;
                                        n2 = lrc->n;

                                        desconecta_DOIS(n2);
                                        transfere_conexao(n1,n2);
                                        libera_no(n2);

                                        lr->l = lrc->l;
                                        libera_no_lista(lrc);
                                    }
                                    libera_no_lista(lrc);

                                    if(n1->at.m.el != n1->at.m.th)
                                    {
                                        for(lf = l; lf->l != NULL; lf = lf->l);
                                        lf->l = cria_no_lista();
                                        lf = lf->l;
                                        lf->n = n1;
                                    }

                                    mudou = 1;
                                    saida = 1;
                                }
                                break;
                            }
                            if(saida)
                                break;
                        }
                        if(saida)
                            break;
                    }
                }
                while(saida);
            }
        }
        while(mudou);
```

```
990
991             laux = l->l;
992             libera_no_lista(l);
993             l = laux;
994         }
995         libera_no_lista(l);
996     }
997
998     QDD* copia_QDD(QDD *Q1)
999     {
1000        /**  l1 guarda Q1 elistado   ***
1001        ***  l2 lista dos nos de l2  **/
1002        lista *l1, *l2, *lc1a, *lc2a, *lc1b, *lc2b;
1003        no *n1, *n2, *nf, *nt1, *nt2;
1004
1005        l1 = enlista_QDD(Q1);
1006        l2 = copia_lista_sem_cabeca(l1);
1007        for(lc2a = l2; lc2a != NULL; lc2a = lc2a->l)
1008            lc2a->n = copia_no(lc2a->n);
1009
1010        lc1a = l1;
1011        lc2a = l2;
1012        do
1013        {
1014            /**  lc1a perocrre a lista 1 pela primeira vez  ***
1015            ***  lc2a percorre a lista 2 pela primeira vez  **/
1016            n1 = lc1a->n;
1017            n2 = lc2a->n;
1018
1019            switch(n1->tipo)
1020            {
1021                case 0:
1022                nf = n1->at.i.n;
1023
1024                lc1b = l1;
1025                lc2b = l2;
1026                do
1027                {
1028                    /**  lc1a perocrre a lista 1 pela segunda vez para buscar filhos
1029                    ***  lc2a percorre a lista 2 pela segunda vez para buscar filhos
1030                    **/
1031                    nt1 = lc1b->n;
1032                    nt2 = lc2b->n;
1033
1034                    lc1b = lc1b->l;
1035                    lc2b = lc2b->l;
1036                }
1037                while(nt1 != nf);
1038
1039                conecta_UM(n2,nt2,0);
1040                break;
1041
1042                case 1:
1043                nf = n1->at.m.el;
1044
1045                lc1b = l1;
1046                lc2b = l2;
1047                do
1048                {
1049                    nt1 = lc1b->n;
1050                    nt2 = lc2b->n;
1051
1052                    lc1b = lc1b->l;
1053                    lc2b = lc2b->l;
1054                }
```

```
1054                while(nt1 != nf);
1055
1056                conecta_UM(n2,nt2,1);
1057
1058                nf = n1->at.m.th;
1059
1060                lc1b = l1;
1061                lc2b = l2;
1062                do
1063                {
1064                    nt1 = lc1b->n;
1065                    nt2 = lc2b->n;
1066
1067                    lc1b = lc1b->l;
1068                    lc2b = lc2b->l;
1069                }
1070                while(nt1 != nf);
1071
1072                conecta_UM(n2,nt2,2);
1073                break;
1074            }
1075            lc1a = lc1a->l;
1076            lc2a = lc2a->l;
1077        }
1078        while(lc1a != NULL);
1079
1080        QDD *Q2;
1081        Q2 = cria_QDD();
1082        Q2->nqbit = Q1->nqbit;
1083        Q2->n = l2->l->n;
1084
1085        lc2a = l2;
1086        while(lc2a->l != NULL)
1087        {
1088            lc2b = lc2a->l;
1089            if(lc2b->n->tipo == 2)
1090                lc2a = lc2b;
1091            else
1092            {
1093                lc2a->l = lc2b->l;
1094                libera_no_lista(lc2b);
1095            }
1096        }
1097
1098        Q2->l = l2->l;
1099        libera_no_lista(l2);
1100        libera_lista(l1);
1101
1102        return Q2;
1103    }
1104
1105    QDD* produto_tensorial(QDD *Q1, QDD *Q2)
1106    {
1107        QDD *Q;
1108        lista *l;
1109
1110        Q = copia_QDD(Q1);
1111        reduz_QDD(Q);
1112        Q->nqbit = Q1->nqbit + Q2->nqbit;
1113        l = Q->l;
1114        Q->l = NULL;
1115
1116        QDD *Q2a;
1117        lista *l2, *lc;
1118        Q2a = copia_QDD(Q2);
1119        reduz_QDD(Q2a);
```

```
1120      l2 = enlista_QDD(Q2a);
1121      for(lc = l2; lc != NULL; lc = lc->l)
1122          if(lc->n->tipo == 1)
1123              (lc->n->at.m.nivel) += (Q1->nqbit);
1124      libera_lista(l2);
1125
1126      QDD *Q2b;
1127      no *n1, *n2, *naux, *n0;
1128      n0 = cria_no_fim(0,0);
1129
1130
1131      while(l != NULL)
1132      {
1133          n1 = l->n;
1134          if(compara_no_fim(n0,n1))
1135          {
1136              l2 = l->l;
1137              l->l = Q->l;
1138              Q->l = l;
1139              l = l2;
1140          }
1141          else
1142          {
1143              Q2b = copia_QDD(Q2a);
1144              for(lc = Q2b->l; lc != NULL; lc = lc->l)
1145              {
1146                  n2 = lc->n;
1147                  naux = produto_complexo(n1,n2);
1148                  transfere_conexao(naux,n2);
1149                  lc->n = naux;
1150                  libera_no(n2);
1151              }
1152
1153              n2 = Q2b->n;
1154              naux = Q2b->n->l->n;
1155              desconecta_UM(naux,n2);
1156              libera_no(naux);
1157
1158              transfere_conexao(n2,n1);
1159              libera_no(n1);
1160
1161              lc = Q2b->l;
1162              while(lc->l != NULL)
1163                  lc = lc->l;
1164              lc->l = Q->l;
1165              Q->l = Q2b->l;
1166              libera_no_QDD(Q2b);
1167
1168              lc = l->l;
1169              libera_no_lista(l);
1170              l = lc;
1171          }
1172      }
1173
1174      libera_no(n0);
1175      libera_QDD(Q2a);
1176
1177      reduz_lista(Q->l);
1178      reduz_QDD(Q);
1179
1180      return Q;
1181  }
1182
1183  QDD* potencia_tensorial(QDD *Q, Short i)
1184  {
1185      QDD *Qs, *Qaux;
```

```
1186      Short j;
1187
1188      Qs = copia_QDD(Q);
1189      for(j=1; j<i; j++)
1190      {
1191          Qaux = produto_tensorial(Qs,Q);
1192          libera_QDD(Qs);
1193          Qs = Qaux;
1194      }
1195      return Qs;
1196  }
1197
1198  void produto_por_escalar(QDD *Q, float re, float im)
1199  {
1200      no *n;
1201      n = cria_no_fim(re,im);
1202
1203      lista *l;
1204      no *n1, *naux;
1205      for(l = Q->l; l != NULL; l = l->l)
1206      {
1207          n1 = l->n;
1208          naux = produto_complexo(n1,n);
1209          transfere_conexao(naux,n1);
1210          l->n = naux;
1211          libera_no(n1);
1212      }
1213
1214      libera_no(n);
1215  }
1216
1217
1218  no* apply_soma(no *N1, no *N2)
1219  {
1220      apply *a, *ac, *a1, *a2;
1221      a = cria_apply();
1222      a->n1 = N1;
1223      a->n2 = N2;
1224      ac = a;
1225
1226      no *n, *n1, *n2;
1227      Short regra;
1228      while(ac != NULL)
1229      {
1230          n1 = ac->n1;
1231          n2 = ac->n2;
1232
1233          switch(n1->tipo)
1234          {
1235              case 1:
1236              /**  caso n1 não seja número  **/
1237              switch(n2->tipo)
1238              {
1239                  case 1:
1240                  /**  caso n2 não seja número  **/
1241                  regra = 0;
1242                  break;
1243
1244                  case 2:
1245                  /**  caso n2 seja número  **/
1246                  regra = 1;
1247                  break;
1248              }
1249              break;
1250
1251              case 2:
```

```
1252                /**  caso n1 seja número  **/
1253                switch(n2->tipo)
1254                {
1255                    case 1:
1256                    /**  caso n2 não seja número  **/
1257                    regra = 2;
1258                    break;
1259
1260                    case 2:
1261                    /**  caso n2 seja número  **/
1262                    regra = 4;
1263                    break;
1264                }
1265                break;
1266            }
1267            if(regra == 0)
1268            {
1269                if(n1->at.m.nivel < n2->at.m.nivel)
1270                    regra = 1;
1271                if(n1->at.m.nivel > n2->at.m.nivel)
1272                    regra = 2;
1273                if(n1->at.m.nivel == n2->at.m.nivel)
1274                {
1275                    if(n1->tipo < n2->tipo)
1276                        regra = 1;
1277                    if(n1->tipo > n2->tipo)
1278                        regra = 2;
1279                    if(n1->tipo == n2->tipo)
1280                        regra = 3;
1281                }
1282            }
1283
1284            switch(regra)
1285            {
1286                case 1:
1287                break;
1288
1289                case 2:
1290                break;
1291
1292                case 3:
1293                break;
1294
1295                case 4:
1296                break;
1297            }
1298        }
1299    }
1300
1301
1302
1303    QDD* I()
1304    {
1305        QDD *Q;
1306        Q = le_matriz("I1.txt");
1307        reduz_QDD(Q);
1308        return Q;
1309    }
1310
1311    QDD* I_n(Short i)
1312    {
1313        QDD *Q1, *Q2;
1314
1315        Q1 = I();
1316        Q2 = potencia_tensorial(Q1,i);
1317        libera_QDD(Q1);
```

```
1318
1319        return Q2;
1320    }
1321
1322    QDD* H()
1323    {
1324        QDD *Q;
1325        Q = le_matriz("H1.txt");
1326        reduz_QDD(Q);
1327        return Q;
1328    }
1329
1330    QDD* H_n(Short i)
1331    {
1332        QDD *Q1, *Q2;
1333
1334        Q1 = H();
1335        Q2 = potencia_tensorial(Q1,i);
1336        libera_QDD(Q1);
1337
1338        return Q2;
1339    }
1340
1341
1342
1343    void inicia_relatorio_memoria(Short i)
1344    {
1345        print = i;
1346        if(print)
1347        fm = fopen("relatorio_memoria.txt","w");
1348    }
1349
1350    void finaliza_relatorio_memoria()
1351    {
1352        printf("\n\nMemMax  : %d",memMax);
1353        printf("\nMemFinal: %d",mem);
1354        if(print)
1355        {
1356            fprintf(fm,"\n\nMemMax: %d",memMax);
1357            fclose(fm);
1358        }
1359    }
1360
1361
1362
1363    int main()
1364    {
1365        inicia_relatorio_memoria(0);
1366        /***********************************/
1367
1368        QDD *Q1;
1369        Q1 = le_matriz("I2.txt");
1370        reduz_QDD(Q1);
1371        produto_por_escalar(Q1,5,7);
1372        mostra_QDD(Q1);
1373
1374        /***********************************/
1375        finaliza_relatorio_memoria();
1376        return 0;
1377    }
```